



UNIVERSITY OF
CAMBRIDGE

A crash course in L^AT_EX (for linguistics)

COPiL editorial team 2021-3

Ema Banerjee, Alex Cairncross

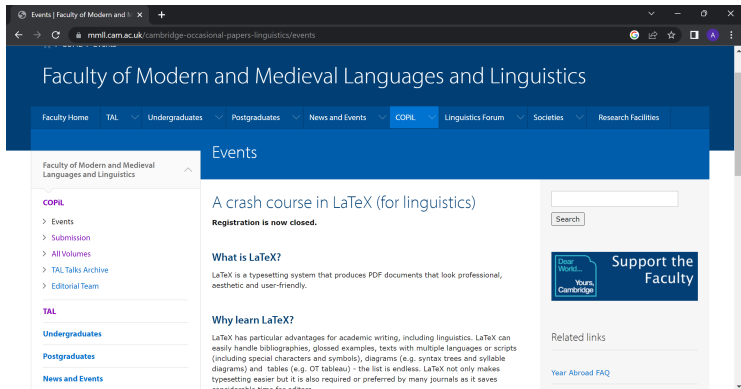
Nina Haket, Sana Kidwai

The University of Cambridge

20 Jan 2023

Want to follow along?

Check email for slides or visit website



The screenshot shows a web browser window with the URL mml.cam.ac.uk/cambridge-occasional-papers-linguistics/events. The page title is "Faculty of Modern and Medieval Languages and Linguistics". The navigation menu includes "Faculty Home", "TAL", "Undergraduates", "Postgraduates", "News and Events", "COPIL", "Linguistics Forum", "Societies", and "Research Facilities". The "COPIL" menu item is active, and a sidebar on the left lists "Events", "Submission", "All Volumes", "TAL Talks Archive", "Editorial Team", "TAL", "Undergraduates", "Postgraduates", and "News and Events". The main content area features the heading "Events" and a sub-heading "A crash course in LaTeX (for linguistics)". Below this, it states "Registration is now closed." and provides information about LaTeX, including "What is LaTeX?" and "Why learn LaTeX?". A search bar and a "Support the Faculty" button are also visible.

Faculty of Modern and Medieval Languages and Linguistics

COPIL

- > Events
- > Submission
- > All Volumes
- > TAL Talks Archive
- > Editorial Team

TAL

Undergraduates

Postgraduates

News and Events

Events

A crash course in LaTeX (for linguistics)

Registration is now closed.

What is LaTeX?

LaTeX is a typesetting system that produces PDF documents that look professional, aesthetic and user-friendly.

Why learn LaTeX?

LaTeX has particular advantages for academic writing, including linguistics. LaTeX can easily handle bibliographies, glossed examples, texts with multiple languages or scripts (including special characters and symbols), diagrams (e.g. syntax trees and syllable diagrams) and tables (e.g. OT tableau) - the list is endless. LaTeX not only makes typesetting easier but it is also required or preferred by many journals as it saves considerable time for editors.

Support the Faculty

Dear World... Yours, Cambridge

Related links

Year Abroad FAQ

Overview

Why L^AT_EX?

Basic document structure

Referencing

Break (20 min)

Floats

Trees

Special characters (IPA / variables)

Glossing

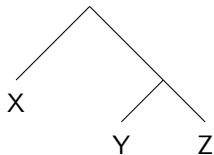
COPiL template

Why L^AT_EX?

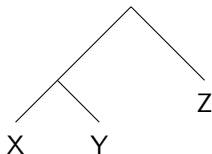
Why L^AT_EX?: Linguistics reasons

Trees

(1)



(2)



Numbered and or glossed examples

- (3) L'=ha mangiat-a Gianni.
CL=has eaten-F Gianni
'Gianni ate it.'

[Italian]

(Cross-)referencing (e.g. 1-3)

Language other than English/ special characters (e.g. è, é, ě, ë, ε, 欸, ə, ∃ ...)

Why L^AT_EX?: Your own sanity

Automate repetitive tasks

- Bibliographies
- List of figures/tables
- Custom commands

Reduce human error

- Referencing
 - o First vs 'et al.'
 - o Missing entries
 - o Alphabetising
 - o Formatting

Formatting and content stored separately

- No more checking for double spaces / unintended indents

Why L^AT_EX?: Other considerations

Ease of customisation

Stack Overflow and Overleaf (docs)

Your poor computer

Stability

Forgetting about formatting

Beyond articles or your thesis

Handouts

- Supervisor meetings
- (Syntax) talks

Posters

- Conference
- Participant recruitment

Presentation slides (like these)

Job applications

- CV
- Cover letter

Caveat

L^AT_EX takes an initial investment

Word/Google Docs has a time and place

- You have an abstract due in an hour and you don't have a template
- Extra features are irrelevant (e.g. COP*i*L meeting notes)
- Some journals require word documents (double check!)
- Your collaborator/supervisor may prefer editable word documents rather than pdfs

Basic Document Structure

Basic Document Structure

Setting up a document

Preamble and packages

Basic formatting

Lists

Sections

Troubleshooting

Setting up a document

Create an Overleaf account (or just log in via Google) if you haven't already.

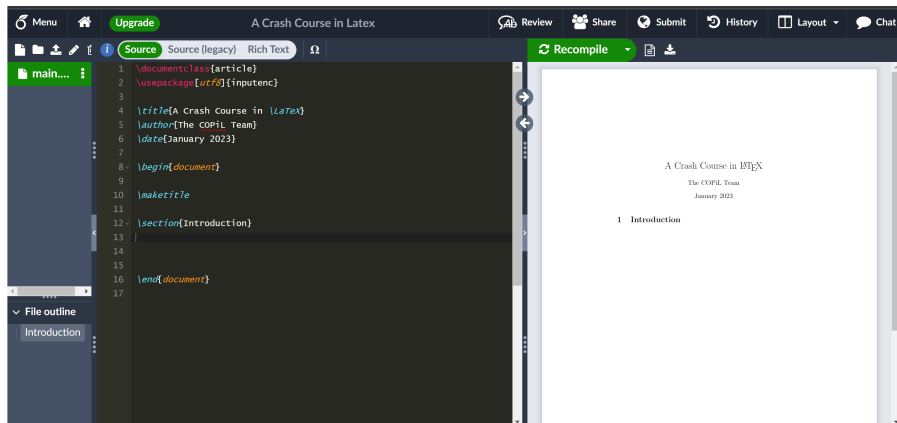
Logging in will take you to the 'Projects' page.

Click *New Project* → *Blank Project*.

Enter a title and create the project.

Setting up a document

Overleaf does a lot of the basic work for you.



The screenshot displays the Overleaf web editor interface. The top navigation bar includes a 'Menu' icon, a home icon, an 'Upgrade' button, the document title 'A Crash Course in LaTeX', and utility buttons for 'Review', 'Share', 'Submit', 'History', 'Layout', and 'Chat'. Below the navigation bar, the editor is split into two main sections. On the left, a dark-themed code editor shows the LaTeX source code for a document. On the right, a light-themed preview window shows the rendered output of the document.

The source code in the left pane is as follows:

```
1 \documentclass{article}
2 \usepackage[utf8]{inputenc}
3
4 \title{A Crash Course in  $\LaTeX$ }
5 \author{The COPiL Team}
6 \date{January 2023}
7
8 \begin{document}
9
10 \maketitle
11
12 \section{Introduction}
13
14
15
16 \end{document}
17
```

The rendered output in the right pane shows the following content:

A Crash Course in \LaTeX
The COPiL Team
January 2023

1 Introduction

The interface also features a 'File outline' panel in the bottom-left corner, which currently shows 'Introduction' as the active section.

Preamble

Preamble: the 'setup' section of a document.

Includes everything before `\begin{document}` (where the actual document contents begins).

Functions:

- Defines document class (article, book, etc).
- Configures the document - languages, page setup, etc.
- Loads packages.
- And lots more! You can create your own commands, define specific colours, make a new environment type, and so on.

Packages

Packages: external bodies of code that provide specialist capabilities or extend \LaTeX 's built in features.

Packages are distributed through CTAN, which currently has 6365 available.

Imported using the command

```
\usepackage[<options>]{<packagename>}
```

Some examples:

- `\usepackage{graphicx}` - for using image files.
- `\usepackage[english,italian]{babel}` - for using (multiple) languages/alphabets.
- `\usepackage[table,dvipsnames]{xcolor}` - for using colours, including by name and in tables.

Basic formatting: formatting text

Bold, italics, underlining, superscript, subscript, small caps.

Commands:

- `\textbf{}`
- `\textit{}` or `\emph{}`
- `\underline{}`
- `\textsuperscript{}`
- `\textsubscript{}`
- `\textsc{}`

Basic formatting: formatting text

```
1 Text can easily be \textbf{bolded},
2 \textit{italicised}, and \underline{underlined}.
3 You can also add superscript text
4 \textsuperscript{like this}, subscript text
5 \textsubscript{like this}, or text in \textsc{small
6 caps}.
```

Text can easily be **bolded**, *italicised*, and underlined. You can also add superscript text ^{like this}, subscript text _{like this}, or text in SMALL CAPS.

Basic formatting: font sizes

There are ten default font sizes: `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge` and `\Huge` (note that capitalisation matters!).

These sizes are relative to the base fontsize (which can be defined in the preamble) - you can also use absolute sizes if you prefer.

They affect whatever text follows them, until another command or a different environment is defined.

```
1 \Huge Very big text, \Large large text, \small  
2 small text, \tiny tiny text.
```

Very big text, large text, small text, tiny text.

Basic formatting: paragraphs and spaces

Paragraphing and other spacing can be done in a few ways.

Line breaks can be inserted by leaving a blank line, typing `\\`, or using `\newline`.

Vertical spaces can be added using `\vspace{<length>}`, or the commands `\smallskip`, `\medskip`, and `\bigskip`.

Horizontal spaces can be added using `\hspace{<length>}`, or `\hfill` to move everything to the other side of the line.

Basic formatting: paragraphs and spaces

```
1 This is a paragraph. \\
2 This is a line with a comment. \hfill This is the
3 comment.
4 \vspace{2cm} This is a line 2cm below the last one.
```

This is a paragraph.

This is a line with a comment.

This is the comment.

This is a line 2cm below the last one.

Lists

Example of an *environment*, which applies specific typesetting effects to just that part of the document. Environments are contained in `\begin{<name>}` and `\end{<name>}` tags.

Unordered (bullet point) lists use `itemize`, ordered (numbered) lists use `enumerate`.

Individual items use `\item`.

Can be nested to create sublists.

Environments must be closed in relative order; most recently opened first.

Lists

```
1 \begin{itemize} \item An item in an unordered list.  
2 \item Another item, introducing a numbered  
3 sub-list.  
4 \begin{enumerate} \item The first item in this  
5 sub-list.  
6 \item A second item.  
7 \end{enumerate} \end{itemize}
```

- An item in an unordered list.
- Another item, introducing a numbered sub-list.
 1. The first item in this sub-list.
 2. A second item.

Lists: examples

`enumerate` restarts the numbering every time...not great for linguists.

Solution: packages! There are several specific packages for formatting examples (with continued numbering, sub-parts, etc.) including `linguex` and `expex`.

Packages that try and do the same thing don't work nicely together - you should stick to one or another.

```
1 \pex \a This is the first sentence.  
2 \a This is another sentence in the same example.  
3 \xe
```

- (4) a. This is the first sentence.
- b. This is another sentence in the same example.

Sections

Sections (and subsections, subsubsections etc) divide documents.

There are multiple levels of depth, depending on the document class.

Sections do a lot of things automatically:

- Numbering (and cross-referencing)
- Formatting headers
- Creating a contents page (using `\tableofcontents`)

`\section{Title}` creates a section, `\subsection{Title}` and `\subsubsection{Title}` create nested (sub)subsections.

Unnumbered sections can be created by adding an asterisk at the end of the command.

Sections

```
1 \section{Section One}
2 \subsection{A subsection}
3 This is just some generic text in the subsection,
4 enough to fill a line or so.
5 \subsection*{Another, unnumbered subsection}
6 Some more text in another subsection.
```

1 Section One

1.1 A subsection

This is just some generic text in the subsection, enough to fill a line or so.

Another, unnumbered subsection

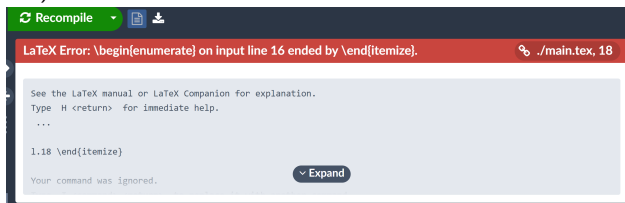
Some more text in another subsection.

Troubleshooting

If there's an issue with the code, \LaTeX can usually figure out what it is (or at least give you the line where it occurs).



Issues can either be warnings (shown in orange, usually auto-corrected, can generally be ignored) or actual errors (shown in red, can prevent compiling, should be fixed).



Other resources: [Overleaf documentation](#), [StackExchange](#), Google.

Basic document structure: your turn

Create some sections (and subsections if you want!) in your document, for example the sections in today's presentation (this section, referencing, floats, trees, special characters, glossing).

- `\section{...}`

Write some text in your introduction and format it.

- e.g. `\textbf{...}`, `\textit{...}`, etc.

Summarise the material in this section using a list.

- `\begin{itemize}...` or `\begin{enumerate}...` (don't forget to end the environment).

Import a package; you can pick one from CTAN, or add the `graphicx` package.

- `\usepackage[<options>]{<packagename>}`

Referencing

Referencing

Cross-referencing

External links

Bibliography

Basic cross-referencing

Referencing sections, examples, tables, figures etc. in the document.

Commands:

- `\label{<label>}`
- `\ref{<label>}`

```
1 \section{Section 1} \label{sec1}
2
3 A reference to section \ref{sec1}.
```

1 Section 1

A reference to section 1.

hyperref package

The `hyperref` package creates hyperlinks.

Preamble:

```
- \usepackage{hyperref}
- \hypersetup{
  colorlinks=true, (links will be coloured, default is red)
  linkcolor=blue, (colour for internal links)
  urlcolor=blue, (colour for URLs)
  citecolor=blue, (colour for citations)
  allcolors=blue, (colour for all types of links)
}
```

Cross-referencing with `hyperref`

Commands:

- `\ref{<label>}`
- `\autoref{<label>}`

```
1 \section{Section 1} \label{sec1}
2
3 A reference to \ref{sec1} vs. \autoref{sec1}.
```

1 Section 1

A reference to [1](#) vs. [section 1](#).

External links with hyperref

URLs:

- `\url{<url>}`
- `\href{<url>}{<text>}`

```
1 \url{https://www.mml1.cam.ac.uk/cambridge-occasional-  
2 papers-linguistics}  
3  
4 \href{https://www.mml1.cam.ac.uk/cambridge-occasional  
5 -papers-linguistics}{The COPiL website}
```

<https://www.mml1.cam.ac.uk/cambridge-occasional-papers-linguistics>

[The COPiL website](#)

External links with `hyperref`

Email addresses:

- `\href{mailto:<email address>}{<text>}`

```
1 Click \href{mailto:copil@mml1.cam.ac.uk}{here} to  
2 email the COPiL team!
```

Click [here](#) to email the COPiL team!

Bibliography

L^AT_EX can automatically format citations, both in-text and in the bibliography.

The bibliography information is stored in a `.bib` file.

The entries in your `.bib` file can be referenced in the document.

Opening a `.bib` file:

- Click *New File* (left-most icon under *Menu*).
- Enter a file name.
- Change the file extension to `.bib`.

.bib file

Each bibliography entry is stored as a .bib entry in the following format:

```
@article{key, (@entry type, key = label)
  author = {},
  title = {{}}, (double brackets to protect caps, also for booktitles)
  journal = {},
  year = {},
  volume = {},
  number = {},
  pages = {},
  note = {},
}
```

A useful masterlist: <https://www.bibtex.com/e/entry-types/>

Bibliography

Preamble:

- `\usepackage{natbib}`
- `\bibliographystyle{apastyle}` (to set the referencing style)

Document: `\bibliography{<bib file name>.bib}`
(to insert the bibliography, usually near the end of the document)

As a default, only the entries you cite in-text will be printed in the bibliography.

In-text citations

Commands:

- `\citet{<key>}`
- `\citep{<key>}`

You can find the rest of the commands in the `natbib` package documentation.

```
1 \citet{Chomsky1980}
2 \citep{Chomsky1980}
```

Chomsky (1980)
(Chomsky, 1980)

Note: If you are using the `hyperref` package, all in-text citations will automatically link to the full reference in the bibliography.

In-text citations

To add information to an in-text citation:

```
\citep[<before>] [<after>] {<key>}
```

```
1 \citep[see] [p.1] {Chomsky1980}  
2 \citep[p.1] {Chomsky1980}  
3 \citep[see] [] {Chomsky1980}
```

(see Chomsky, 1980, p.1)

(Chomsky, 1980, p.1)

(see Chomsky, 1980)

Referencing: your turn

1 First Section

In this section, we're going to practise referencing (1).

1. Submit to [COPiL](#)!

[Chomsky](#) (1980, p.1-5) said some stuff (see also [Chomsky, 1981, 1991](#)). All of Chomsky's ideas are discussed in [section 2](#).

2 Second Section

Something about linguistics.

References

Chomsky, N. (1980). On Binding. *Linguistic Inquiry*, 11(1):1–46. <https://www.jstor.org/stable/4178149>.

Chomsky, N. (1981). *Lectures on Government and Binding*. Walter de Gruyter & Co., Berlin.

Chomsky, N. (1991). Some Notes on Economy of Derivation and Representation. In Freidin, R., editor, *Principles and Parameters in Comparative Grammar*, pages 417–454. MIT Press, Cambridge, MA.

Break

Floats

Floats

Unit that should stay together, but maybe not right here

- Tables
- Figures
- Custom

Separation of formatting and content

General pattern: top, bottom, next top

Import an image

Package: `graphicx`

Command: `\includegraphics{<path to file>}`

Option: specify width/height in your favorite unit (cm, in, % of `textwidth` ...)

```
1 \includegraphics[width=.35\textwidth]{Images/Pro-  
2   crastination.jpg}
```



Images as figures

Standalone image is not nicely formatted not useful (no caption / no crossref)

Solution: Figure environment

```
\begin{figure}[<position>]
  \centering
  \includegraphics[...]{...}
  \caption{<caption>}
  \label{<label>}
\end{figure}
```

Basic positions: t[op], b[ottom], h[ere]

Caption must proceed label!

Tip: indent for future you

Example figure

```
1 \begin{figure}
2   \centering
3   \includegraphics[width=.5\textwidth]{Images/...}
4   \caption{Me on a daily basis}
5   \label{Me}
6 \end{figure}
```

Example figure



Figure 1: Me on a daily basis

Tabular environment

tabular \neq table

- tabular environments create collections of cells ← ‘includegraphics’
- Tables (containing tabulars) float ← ‘figure’

Basic tabular case

```
\begin{tabular}{<columns>}
  <contents> & <contents> \\
  <contents> & <contents> \\
\end{tabular}
```

column alignments: c[enter], r[right], l[left]

lines (for top/bottom/midrule: `\usepackage{booktabs}`)

- vertical: | in column specifications
- horizontal: `\toprule`, `\bottomrule`, `\midrule` in between rows

Tabular with rule lines

```
1 \begin{tabular}{r|cc}
2   \toprule
3       & Control & Experimental \\
4   \midrule
5   Condition A & 10 & 15 \\
6   Condition B & 17 & 16 \\
7   \bottomrule
8 \end{tabular}
```

	Control	Experimental
Condition A	10	15
Condition B	17	16

Tabular and merging

Package: multicol

Merge horizontally: `\multicolumn{<# cols>}{<align>}{<content>}`

Merge vertically: `\usepackage{multirow}` ← preamble
`\multirow{<# rows>}{<width>}{<content>}` ← '*' = inherit width

Merge both?: you can nest multi row in multi column ...but not often required

Wrap table environment like figure to make float

Tabular multicol and cmidrule example

```
1 \begin{table}
2 \centering
3 \begin{tabular}{r cc}
4 \toprule
5 & \multicolumn{2}{c}{Control} \\
6 \cmidrule{2-3}
7 & Mean & SD \\
8 \midrule
9 Condition A & 10 & 3 \\
10 Condition B & 17 & 4 \\
11 \bottomrule
12 \end{tabular}
13 \caption{Some data}
14 \end{table}
```

Tabular multicol and cmidrule example again

	Control	
	Mean	SD
Condition A	10	3
Condition B	17	4

Table 1: Some data

Table Generators

The screenshot shows the TablesGenerator.com web interface. At the top is a menu bar with 'File', 'Edit', 'Table', 'Column', 'Row', 'Cell', and 'Help'. Below the menu is a toolbar with icons for table styles, bold, italic, underline, grid, and other table manipulation functions. The main area contains a table editor with columns labeled A, B, C, D, E and rows numbered 1 to 4. The first cell (A1) is highlighted in yellow. Below the editor is a 'Generate' button. The 'Result' section shows the following LaTeX code:

```
1 \begin{table}[]
2 \begin{tabular}{l11111}
3 & & & & \\
4 & & & & \\
5 & & & & \\
6 & & & & \\
7 \end{tabular}
8 \end{table}
```

At the bottom, there is a checkbox labeled 'Escape special TeX symbols (% , & _ #, \$)' which is checked.

Figure 2: www.tablesgenerator.com

OT tableaux

```
Define hand: \usepackage{pifont}  
\newcommand{\hand}{\ding{43}}
```


/stap/	*COMPLEX	ANCH.-IO	CONT.-IO
a. stap	*!		
 b. sap			*
c. tap		*!	

Figure 3: Example of an OT tableau

Basic tabular is a bit tedious ...

Instead use a custom package like [ot-tableau](#) (used to make figure above)

OT tableaux again

```
1 \ShadingOn
2 \begin{tableau}{c:c|c}
3 \inp{\ips{stap}}
4 \const{*Complex} \const{Anch.-IO} \const{Cont.-IO}
5 \cand{stap}          \vio{*!} \vio{} \vio{}
6 \cand[\Optimal]{sap} \vio{} \vio{} \vio{*}
7 \cand{tap}          \vio{} \vio{*!} \vio{}
8 \end{tableau}
```

Floats: your turn

(α) Insert your favourite meme \leftarrow `\includegraphics{}`

(β) Make it into a figure with a caption and label \leftarrow `\begin{figure}`
`...`
`\end{figure}`

(γ) Cross-reference it in the running text somewhere \leftarrow `\autoref{}`

(δ) Then pick 1 (depending on applicability)

- Make a table with horizontally merged cell \leftarrow `\multicolumn{ }{ }{ }`

- Make an OT tableau with the ot-tableau pack. \leftarrow `\begin{tableau}`
`...`
`\end{tableau}`

Trees

Trees

There are lots of packages that can make pretty trees - today we'll be using `forest`.

`forest` has a lot of customisation options; as with most packages, the details of its functionality are available in its documentation (hosted on CTAN too, click [here](#)).

`qtree` is another popular option.

Trees can be implemented within figures or examples (e.g. using the `expex` package).

Unlike images (of trees), these can be changed and customised, and adhere to global formatting (e.g. font).

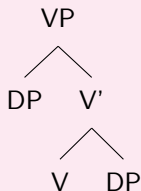
Trees: the forest package

Preamble: `\usepackage[linguistics]{forest}` (the `linguistics` option formats standard syntax trees well).

Environment: `\begin{forest}...\end{forest}`

Content: `forest` uses (labelled) square brackets as its input.

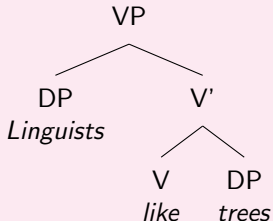
```
1 \begin{forest} [VP [DP] [V' [V] [DP]]] \end{forest}
```



forest: formatting

You can format within nodes the same way you would normal text - e.g. new lines or italics.

```
1 \begin{forest} [VP [DP \\ \textit{Linguists}]
2 [V' [V \\ \textit{like}] [DP \\ \textit{trees}]]]
3 \end{forest}
```



forest: formatting

There are plenty of other formatting options: distance between nodes, alignment, tree direction, line thickness, fonts, colours, borders...

Some example commands:

- `calign = ordinal number` changes which child is aligned with parent
- `grow=direction` changes the direction of the tree
- `draw` draws (customisable) borders around each node
- `inner sep = length` sets size of node
- `s sep = length` changes distance between siblings
- `l sep = length` changes distance between parent and child
- `fill = colour` fills nodes with colour

forest: formatting

If these options are specified within a node, they will only apply to that node.

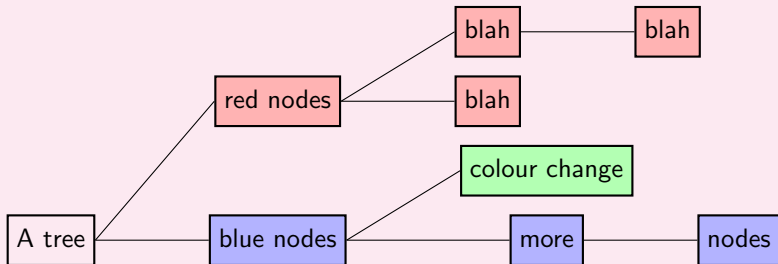
If you want them to more of the tree, we need to *propagate* them, using the command `for tree={options}`.

If `for tree` goes directly after `\begin{forest}`, the options will apply to the whole tree.

Otherwise, you can use `for tree` in a node, and it will apply to that node and all its descendants.

forest: formatting

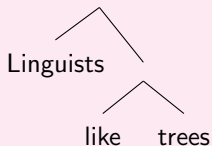
```
1 \begin{forest} for tree={grow=east, calign=first,  
2 draw={black,thick}, l sep=1.5cm} [A tree  
3 [blue nodes, for tree={fill=blue!30}  
4 [more[nodes]][colour change, fill=green!30]][red  
5 nodes, for tree={fill=red!30} [blah][blah  
6 [blah]]]]\end{forest}
```



forest: empty nodes

Empty nodes are useful but they don't look very nice when done automatically.

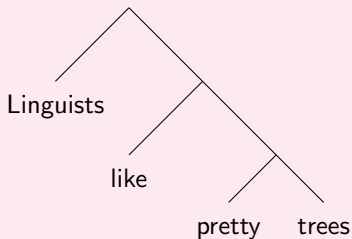
```
1 \begin{forest} [[Linguists] [ [like] [trees]]]  
2 \end{forest}
```



forest: empty nodes

We can use `nice empty nodes` to improve this.

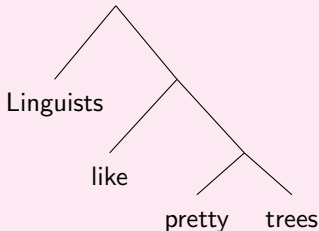
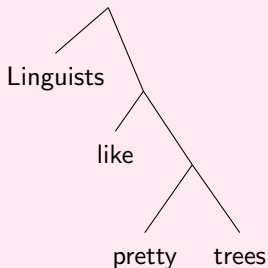
```
1 \begin{forest} [, nice empty nodes [Linguists] [  
2 [like] [trees]]] \end{forest}
```



forest: empty nodes

There are also variants called `pretty nice` and `fairly nice` (you may have to copy the code for these separately though).

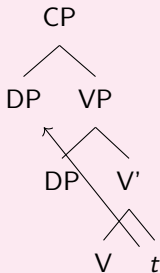
```
1 \begin{forest}[, pretty/fairly nice empty  
2 nodes[Linguists][[like][trees]]\end{forest}
```



forest: arrows

To add arrows to trees, we have to label the relevant nodes (using `name=label`), and then draw a line between them, using `\draw [->]` (`<source>`) to [`<options>`] (`<target>`).

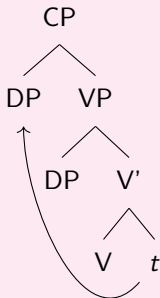
```
1 \begin{forest} [CP [DP,name=spec CP] [VP [DP ] [V'  
2 [V] [\textit{t},name=object]]]]  
3 \draw[->] (object) to (spec CP); \end{forest}
```



forest: arrows

We need our arrow to curve - this is when the [options] come in handy.

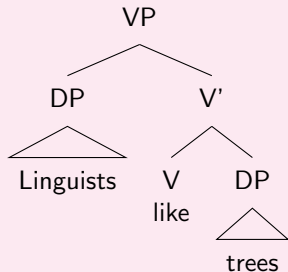
```
1 \begin{forest} [CP [DP,name=spec CP] [VP [DP ] [V'  
2 [V] [\textit{t},name=object]]]]  
3 \draw[->] (object) to[out=south west, in=south]  
4 (spec CP); \end{forest}
```



forest: triangles

Triangles are also very easy in `forest`, using the `roof` command.

```
1 \begin{forest}
2 [VP [DP [Linguists, roof] ]
3 [V' [V \\ like ] ] [DP [trees, roof]]
4 \end{forest}
```



forest: beyond syntax

Syntax trees are far from the only thing you can use `forest` for.

More generally: flowcharts, decision trees, etc.

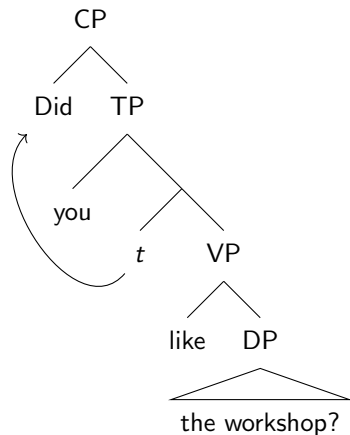
Computational linguistics diagrams, e.g. Finite State Automata/Transducers, Push-Down Automata.

Phonology, e.g. Government Phonology (including the GP1 style).

Other linguistics: language families, etymologies, diachronic change representations...

Trees: your turn

Try to recreate the following (or a similar sentence):



Note how `nice` those empty nodes look.

The distance between parents and children is 1.5cm (`1 sep=...`).

The arrow goes both out and in at the southwest of its nodes.

Node options (like names and roofs) go after the node content, separated by commas (`[node name, option 1, option 2]`).

Special Characters

Special Characters

Special characters are anything that goes beyond what you have on your keyboard

In this case, we will focus on the IPA, mathmode, and semantics symbols

And also introduce you to detexify to help you get the code you need for each symbol quicker

IPA and `tipa`

The package needed for the IPA is called `tipa`

Preamble: `\usepackage{tipa}`

As with other topics seen, this introduces new characters, environments and commands

Why `tipa`?

A new 256 character encoding for phonetic symbols which includes all the symbols and diacritics found in the recent versions of IPA and some non-IPA symbols.

Easy input method in the IPA environment

Extended macros for accents and diacritics

A flexible system of macros for 'tone letters'

An optional package (`vowel.sty`) for drawing vowel diagrams

tipa Characters

Every TIPA phonetic symbol has 3 things:

A unique symbol name, such as Turned A, Hooktop B, Schwa.

A corresponding control sequence, or macro, name, such as `\textturna`, `\texthtb`, `\textschwa`.

And many have a shortcut character that refers to a single character that is assigned to a specific phonetic symbol and that can be directly input by an ordinary keyboard

Macro characters

The name used as a control sequence is usually an abbreviated form of the corresponding symbol name with a prefix `\text...`

With some small changes, such as removal of suffixes, 'l' and 'r' for left and right respectively, sc for small capitals, ht for hooktop, ct for curly tail

Symbol name	Macro name	symbol
Turned A	<code>\textturna</code>	æ
Glottal Stop	<code>\textglotstop</code>	ʔ
Right-tail D	<code>\textrtaid</code>	ɖ
Small Capital G	<code>\textscg</code>	Ꞥ
Hooktop B	<code>\texthtb</code>	Ƀ
Curly-tail C	<code>\textctc</code>	ꞥ
Crossed H	<code>\textcrh</code>	ħ
Beta	<code>\textbeta</code>	β

Shortcut characters

Shortcut	:	;	”							
IPA	ː	ˑ	”							
Shortcut	0	1	2	3	4	5	6	7	8	9
IPA	ʈ	ɨ	ʌ	ɜ	ɥ	e	ɒ	ɹ	ɵ	ə
Shortcut	@	A	B	C	D	E	F	G	H	I
IPA	ə	ɑ	β	ɸ	ð	ε	φ	ɣ	ɦ	ɪ
Shortcut	J	K	L	M	N	O	P	Q	R	S
IPA	ɟ	ɸ	ʎ	ɱ	ŋ	ɔ	ʔ	ʃ	ɾ	ʃ
Shortcut	T	U	V	X	Y	Z	—			
IPA	θ	ʊ	v	χ	ʏ	ʒ	ɿ			

tipa Diacritics and Tone

Almost infinite possibilities, too many for a single slide, so will highlight some.

Macro	IPA environment	Symbol
Accents		
<code>\'a</code>	<code>\'a</code>	á
<code>\"a</code>	<code>\"a</code>	ä
Suprasegmentals		
<code>\textsubbridge{t}</code>	<code>\—[t</code>	t̲
<code>\textsubplus{o}</code>	<code>\—+o</code>	o̰
Tone (<code>\usepackagetone{tipa}</code>)		
<code>\tone{55}ma</code>	<code>\tone{55}ma</code>	ᵇma
<code>\tone{214}ma</code>	<code>\tone{214}ma</code>	ᵇma

Using `tipa`

There are two ways to input phonetic symbols using `tipa`, and the first is inputting macro names or shortcut characters within special environments

Environment : `\begin{IPA} ... \end {IPA}` OR `\textipa{...}`

In these environments you can use the shortcut symbols

```
1 \textipa {"Ekspɫ@neɪs@n}
2 or
3 \begin {IPA}
4     ["Ekspɫ@neɪs@n]
5 \end {IPA}
```

[,ɛksplə'neɪʃən] or [,ɛksplə'neɪʃən]

Using `tipa`

The second way is to enter the macro commands in the normal text environment.

```
1 [\textsecstress \textepsilon kspl\textschwa  
2 \textprimstress ne\textsci \textesh \textschwa n]
```

[,ɛksplə'neɪʃən]

How on earth am I going to remember all of these characters?

Luckily, you don't! All are easily googleable, and many websites have special cheatsheets. For example:

www.tug.org/tugboat/tb17-2/tb51rei.pdf

<https://ptmartins.info/tex/tipacheatsheet.pdf>


BUT there are clear and easy to remember patterns! e.g. if you know what `\textrtaild` (right tailed-d) does, you can probably guess how to get a t with a right tail, or a n with a left tail... that is a huge advantage over ALT codes or word insert feature

Detexify

Another way of accessing the code for all of the IPA symbols, and all the special characters more generally is through Detexify!

All are easily googleable, and many websites have special cheatsheets

Detexify classify symbols



Want a Mac app?

Lucky you. The Mac app is finally stable enough. See how it works on [Vimeo](#). Download the latest version [here](#).

Restriction: In addition to the LaTeX command the unlicensed version will copy a reminder to purchase a license to the clipboard when you select a symbol.

Score: 0.10305042949249144
 $\text{\usepackage{ tipa }}
 \textschwa
textmode$

Score: 0.13694303328531546
 $\text{\usepackage{ amsymb }}
 \backepsilon
mathmode$

Score: 0.14008839505979676
 \supseteq
mathmode

Score: 0.14530390993030218
 \ni
mathmode

Score: 0.15095017847203826
 \exists
mathmode

The symbol is not in the list? [Show more](#)

Did this help?

Mathmode

Good for any equations you may need to use, and also good for many semantic symbols

You can use any of these 'delimiters' to typeset your math in inline mode

`$...$`

`\begin{math}...\end{math}`

Can also have a separate equation on separate line that you can refer back to later using the `\autoref` from earlier!

`\begin{equation}...end{equation}`

Mathmode Characters

Again, too many to mention here, but I can refer you to a really good summary on these websites:

https://psumikeputnam.weebly.com/uploads/4/0/2/1/40212369/mathmode_latex.pdf

https://www.overleaf.com/learn/latex/List_of_Greek_letters_and_math_symbols

Small sample to get you started:

Mathmode Characters

```
1 $\Delta, \lambda$  
2 $a\pm b$  
3 $\sigma$  
4 $\sum \limits_{i=1}^n x_i$  
5 $a\leq b$
```

Δ, λ
 $a \pm b$
 σ
 $\sum_{i=1}^n x_i$
 $a \leq b$

Semantics characters

Most can be done in mathmode, since the majority of semantics symbols are just maths symbols:

<code>\neg</code>	\neg	<code>\in</code>	\in	
<code>\&</code>	$\&$	<code>\subset</code>	<code>\supset</code>	$\subset \supset$
<code>\rightarrow</code>	\rightarrow	<code>\alpha</code>	α	
<code>\exists</code>	\exists	<code>\lambda</code>	λ	
<code>\forall</code>	\forall	<code>\phi</code>	ϕ	

1 `\lambda x. Different_from(x, Sam)`

$\lambda x. \text{Different_from}(x, \text{Sam})$

Semantics characters

There are a few things you do need other packages or methods for, namely DRT and `box/diamond` notation

DRT can be done in a `tabular` environment or `array`

For `box/diamond` notation, you'll need the `latexsym` package, and many other less frequent symbols will require the same package:

```
1 \usepackage{latexsym}           ←In the preamble!  
2 \diamond
```



Great big list

<https://mirror.apps.cam.ac.uk/pub/tex-archive/info/symbols/comprehensive/symbols-a4.pdf>

Loads of different characters we haven't had the time to go over!



Special characters: your turn

Can you write your name in IPA using latex?

Can you write Eistein's famous mass-energy equivalence equation?

Glossing

Glossing

There are many packages that can be used for glossing. The one we will look at today is the `expex` package.

`expex` has many customisation options for numbering, spacing, text style, and so on, which are all explained in the package documentation.

Another popular package is `gb4e`. It has fewer customisable options but works fine for most data.

expex package

Preamble: `\usepackage{expex}`

`expex` introduces new environments as well as new commands.

Remember, environments must be opened and closed.

Environment 1: `\ex ... \xe`

```
1 \ex An example \xe
```

(5) An example

expex package

Environment 2: `\pex ... \xe`

```
1 \pex An example with parts
2 \a Part a
3 \a Part b
4 \xe
```

(6) An example with parts

- a. Part a
- b. Part b

Glossing using expex

```
\begin{expex} (this is an environment which needs to be closed)
  \gla <language line> //
  \glb <gloss line> //
  \glc <another gloss line> //
  \glft <translation> //
\end{expex} (close environment)
```

Glossing using expex

```
1 \ex
2 \begin{gls}
3 \gla Yeh Urdu he//
4 \glb This Urdu is//
5 \glft 'This is Urdu.'//
6 \end{gls}
7 \xe
```

(7) Yeh Urdu he.
This Urdu is
'This is Urdu.'

Glossing using expex

```
1 \ex
2 \begin{gls}
3 \gla Yeh Urdu he//
4 \glb Subj Obj V//
5 \glc This Urdu is//
6 \glft 'This is Urdu.'//
7 \end{gls}
8 \xe
```

(8) Yeh Urdu he.
Subj Obj V
This Urdu is
'This is Urdu.'

Glossing using expex

```
1 \ex \ljudge*
2 \beginl
3 \gla Yeh nahi Urdu he//
4 \glb This not Urdu is\textsc{.3sg}//
5 \glft 'This is not Urdu.'//
6 \endgl
7 \xe
```

(9)*Yeh nahi Urdu he.
This not Urdu is.3SG
'This is not Urdu.'

Glossing: your turn

Gloss your own language examples in \LaTeX . Try to produce a multi-part example.

The COP*i*L template

Useful links

[CTAN](#): repository for packages (and their documentation).

[Overleaf documentation](#): comprehensive help for everything relating to Overleaf.

[StackExchange](#): Q+A forum.

[BibTeX website](#): guides and tips to using BibTeX.

[Tables Generator](#): create (or import from Excel) tables.

[Detexify](#): draw symbols to find their code/necessary packages.

Character cheat-sheets for [TIPA](#) and for [mathmode symbols](#).

[COPiL submission page](#): guidelines for submission and template.

End